İSTANBUL UNIVERSITY CERRAHPAŞA

ORIGINAL ARTICLE
Electrica 2023; 23(3): 634-642

IoT Malware Detection Based on OPCODE Purification

İbrahim Gülataş[1], Hacı Hakan Kılınç[2], Muhammed Ali Aydın[3], Abdul Halim Zaim[4]

[1]Department of Computer Engineering, Science Institute of Istanbul Commerce University, İstanbul, Turkey
[2]Department of Research and Development, NetRD Information Technologies and Telecommunications, İstanbul, Turkey
[3]Department of Computer Engineering, Istanbul University-Cerrahpasa, Faculty of Engineering, İstanbul, Turkey
[4]Department of Computer Engineering, Istanbul Commerce University, İstanbul, Turkey

Cite this article as: İ. Gülataş, H.H. Kılınç, M.A. Aydın and A.H. Zaim, "IoT malware detection based on OPCODE purification," *Electrica*, 23(3), 634-642, 2023.

ABSTRACT

Malware threat for Internet of Things (IoT) devices is increasing day by day. The constrained nature of IoT devices makes it impossible to apply high-resource-demand ing anti-malware tools for these devices. Therefore there is an enormous need for lightweight and efficient anti-malware solutions for IoT devices. In this study, machine learning-based malware detection is performed using purified OPCODE analysis for IoT devices with MIPS architecture. The proposed methodology reduced the runtime of IoT malware detection up to 7.2 times without reducing the accuracy ratio.
Index Terms—Internet of Things Malware detection, malware analysis, Operation Code analysis

I. INTRODUCTION

Internet of Things (IoT) devices are one of the latest trends in information technologies with a widespread area of utilization. The widespread usage of these devices also takes attention of the malicious users and organizations. The small size and mobility of these devices also yield some constraints in terms of computation power, memory, and power consumption. This constrained nature of IoT devices makes them easy targets for malicious activities. The cyber security company SonicWall reports that IoT malware attacks increased by 123% in healthcare during 2022 [1]. It is also indicated in the report that IoT malware attacks for mining cryptocurrencies on compromised devices also increased by 11%. This increase in malware activities for IoT devices is not limited to the healthcare industry and cryptocurrency mining activities, many industries have seen significant increases in IoT malware attacks. With the Covid-19 pandemic, this rate has increased even more. Another cyber security company Zscaler reported that malware attacks increased by 700% during the pandemic [2].

The infamous Mirai malware showed the importance of IoT malware attacks. Mirai gains initial access to the victim device by brute forcing default credentials. The infected devices are used for creating a botnet to initiate DDoS (Distributed Denial of Service) attacks. Due to the number of infected devices, the impact of the attack was devastating. During the peak of the attack, 600 Gbps is generated and this volume was the record of generated data traffic on a single attack [3]. The least powerful devices caused the biggest DDoS attack. Unfortunately, Mirai is not the last example of IoT malware and the attack DDoS types are not limited to DDoS attacks. In our previous research, we analyzed the malware threat for IoT devices and we were able to locate 64 malware and 43 of them were developed after the Mirai [4]. In that research, we also highlighted that the attack types are not limited to DDoS attacks but also mining cryptocurrencies, and DNS Poisoning attacks are in scope.

Due to the constrained nature of IoT devices, high-resource-demanding anti-malware solutions are not applicable to those devices. Lightweight anti-malware tools are needed for securing those devices against malware attacks. Therefore hash-based solutions are not practical due to high memory requirements. There are several ML (machine learning)–based research in the literature and most of these studies are based on static analysis of malware. Nevertheless, there is no matured anti-malware tool for IoT devices yet.

Corresponding author:
İbrahim Gülataş

E-mail:
ibrahim.gulatas@istanbulticaret.edu.tr

Received: March 20, 2023

Accepted: May 30, 2023

Publication Date: August 1, 2023

DOI: 10.5152/electrica.2023.23043

This study presents the preliminary results of our efforts for developing lightweight IoT malware detection based on OPCODE purification and binary analysis. In this research, binary and static analysis techniques are applied to malware and benignware. There are various CPU architectures for IoT devices such as ARM, Intel, MIPS, Sparc, Motorola, etc. Malware and benignware which are compiled for MIPS architecture analyzed in the scope of this research. The extracted OPCODEs are grouped in accordance with the MIPS instruction set. By this means, we reduced the number of features in our dataset. Finally, we applied ML classifiers to our dataset. The results are awe-inspiring. The Training and Classification times are substantially reduced while there is a slight difference in performance metrics.

The main motivation behind this research is to propose a lightweight malware classification method for IoT devices. In order to achieve this goal, the OPCODEs of malware and benignware samples are grouped by their functions. As a result of this efforts, we obtained a big decrease in the number of features in our dataset. With this new dataset, we are able to reduce the runtime of ML classifiers while the other evaluation metrics are almost the same. The main contribution of this research is summarized below.

- OPCODE purification method is proposed to develop a lightweight IoT malware detection mechanism based on static OPCODE analysis .

This article is organized as follows. Section II gives brief information about recently conducted studies in this domain. Our methodology and dataset creation is given in Section III. Section IV discusses the results of the analysis and classification models. Finally, Section V concludes the article.

II. RELATED WORK

The effects of IoT malware may become devastating; however, efforts to withstand this threat are not satisfying. Even though IoT devices are far from assessed as being resistant to malware attacks, there are some invaluable efforts to secure these devices.

Most of these efforts utilize ML classification techniques based on static analysis of IoT malware. Ngo et al. reviewed 10 previously published papers and defined the features used in those studies [5]. They also compared the results of these studies with their dataset. It has been found that non-graph-based approaches have more successful results. Darabian et al. created a corpus containing the frequencies of the OPCODEs [6]. They used Support Vector Machine (SVM), Random Forest, Decision Tree, K-Nearest Neighbors (KNN), Adaboost, and Multilayer Perceptron (MLP) algorithms and achieved an accuracy rate of over 99%. Su et al. [7] formatted the binary files as 8-bit sequences and then converted them into gray-scale images. They made classification with Convolutional Neural Network (CNN) on the image files they created and obtained 94% accuracy. Azmoodeh et al. [8] used a dataset containing 128 ARM-based malware and 1078 benignware and created a Control Flow Graph (CFG) with OPCODEs. Deep Eigenspace was used for classification and a 98% accuracy rate was obtained. Haddad Pajouh et al. [9] used a dataset containing TF-IDF and frequency values of OPCODEs that belong to 281 ARM-based malware and 270 benignware. They used long short-term memory recurrent neural networks for classification and achieved 98% accuracy. After 2 years, the authors used the same dataset, Gray Wolf Optimization for feature selection and SVM for classification [10]. They achieved an accuracy rate of 99.72%. Alasmary et al. [11] focused on both IoT and Android malware detection. They collected 2962 malware for IoT devices. They applied the CFG model to the dataset with Logistic Regression, SVM, Random Forest, and CNN classification algorithms. As a result, they achieved 99.66% accuracy with CNN.

Also, there are a few research that uses dynamic analysis to train ML classifiers. Meidan et al. achieved 100% TPR (True Positive Rate) and 1% FPR (False Positive Rate) by utilizing network traffic data [12]. Jeon et al. [13] converted behavioral data of IoT malware into image files to detect IoT malware and they reached an accuracy rate of 99.28%. Rey et al. conducted one of the latest research in this field [14] with network traffic data. In their research, they applied two different models. Multilayer Perceptron is applied as a supervised model and achieved a 99.38% accuracy rate. For the unsupervised model, artificial neural network (ANN) was applied and 99.98% TPR rate and 91.78% TNR are achieved.

III. METHODOLOGY

The proposed methodology of this research can be described in four steps. First, malware and benignware for MIPS architecture are collected. Second, static analysis techniques are applied for feature extraction. Thirdly, OPCODE purification is applied by grouping OPCODES by their functions in accordance with the MIPS instruction set to reduce the dimension of the dataset. Finally, the dataset is used for training different ML models to reveal the best-fit ML model. The general overview of our methodology is presented in Fig. 1.

A. Dataset Collection

Developing an efficient malware detection mechanism highly depends on having a comprehensive dataset. Unfortunately, there are only a few publicly available IoT datasets [8, 12, 15–18]. One of the biggest shortcomings of these datasets is most of them are outdated. Due to the fact that IoT malware types are always improving their attack vectors and infection methods, the malware datasets need to be updated regularly. Another shortcoming of these datasets is the lack of benignware in the dataset. Most IoT devices run a specific firmware and except for the Raspberry Pi application store, there are no publicly available tools or files.

In this research, MIPS architecture is chosen as the target architecture because of the widespread area of utilization. We downloaded 212 active IoT malware which are compiled for MIPS architecture, utilizing Malware Bazaar database API [19]. Benignware ELF files are collected from the firmware and update packages of the IoT device vendors. These files are also scanned by VirusTotal for ensuring that they are not used in any malicious activities. As a result of these efforts, we had a collection of 212 malware and 201 benignware ELF files.

B. Static Analysis

The static analysis is conducting reverse engineering methods to extract the behavioral pattern of the malware. In this approach, information about the malware is gathered without running it. Analysis work needs to be done in a private and isolated environment to prevent any infections.

Radare2 is a framework for analyzing and reverse engineering files. This framework is mostly adopted framework among malware researchers. We also utilized the Radare2 framework for conducting static analysis on the collected ELF files. Besides, the Radare2 framework has a Python library (r2pipe) that enables to use of reverse

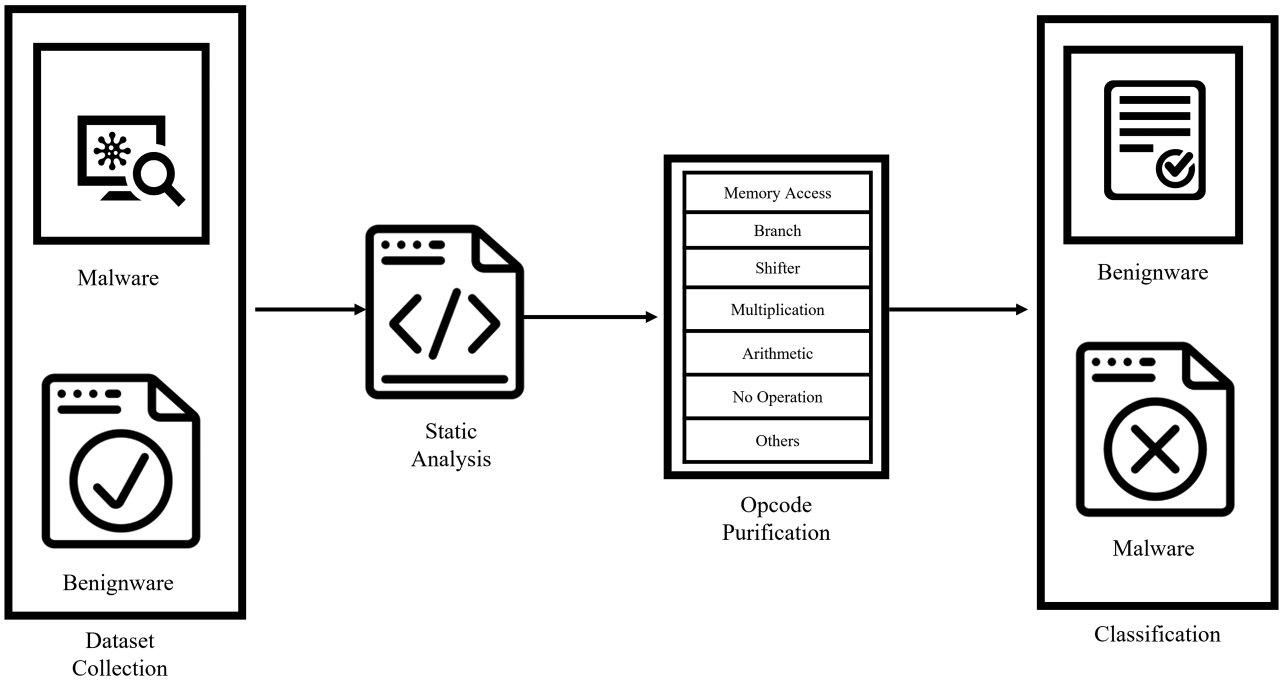

Fig. 1. General overview of methodology.

engineering tools in a Python program and it allows analyzing the ELF files in batches.

Text mining techniques are adopted in the static analysis phase. Initially, we extracted the OPCODEs of the ELF files and copied them to separate text files. Then we calculated the frequencies of OPCODEs in each file and copied them into a .csv file. We located 142 unique OPCODEs that are used in our collected ELF files. We also added three more features from the binary analysis results which are the size of the file, information of being stripped or not, and function number. As a result of these efforts, a dataset is created that contains 413 records with 145 features. This dataset is labeled as *Dataset-1 (DS_1)*. A sample of *Dataset-1* after the binary and static analysis is given in Fig. 2.

C. OPCODE Purification

As it is mentioned in the previous section, our dataset contains 145 features and the size of the dataset causes a high training runtime.

File_Name	Size	Stripped	Function_Count	add	addi	addiu	addu	and	andi	bal	bc0f
./MIPS_Malware/baa2737	23512	0	5	0	0	125	48	2	5	3	0
./MIPS_Malware/3f11d1a	36400	0	8	7	1	136	46	3	6	7	0
./MIPS_Malware/03e5ef7	27420	0	7	7	1	136	46	3	6	7	0
./MIPS_Malware/dea4d37	31268	0	7	7	1	136	46	3	6	7	0
./MIPS_Malware/e072a28	84540	1	4	0	0	97	16	1	5	2	0
./MIPS_Malware/40d22cd	14680	0	5	0	0	125	48	2	5	3	0
./MIPS_Malware/9864cc3	108352	1	4	0	0	84	15	1	4	2	0
./MIPS_Malware/f534d3fa	32736	0	5	0	0	125	48	2	5	3	0
./MIPS_Malware/091a86b	125962	0	215	1	0	2579	502	39	239	4	0
./MIPS_Malware/211ddd1	158537	0	235	1	0	3216	595	41	257	4	0
./MIPS_Malware/4295424	32204	0	7	7	1	136	46	3	6	7	0
./MIPS_Malware/ac04dbc	1156429	0	1527	126	410	99977	21075	2333	7469	2587	3
./MIPS_Malware/0de010a	30724	0	7	7	1	136	46	3	6	7	0
./MIPS_Malware/8d3000b	134222	0	215	1	0	2917	563	38	215	4	0
./MIPS_Malware/23161ee	61491	1	4	0	0	32	6	1	1	2	0
./MIPS_Malware/daa85f3	152836	1	4	0	0	98	14	1	5	2	0
./MIPS_Malware/94a0ad5	28104	0	6	0	0	125	48	2	5	3	0
./MIPS_Malware/ee965ce	76236	1	75	0	0	12	4	1	0	2	0
./MIPS_Malware/3bdcbfe	27880	0	6	0	0	125	48	2	5	3	0
./MIPS_Malware/4f12fe6	32632	0	6	0	0	125	48	2	5	3	0
./MIPS_Malware/e192698	38792	0	7	7	1	136	46	3	6	7	0

Fig. 2. A sample of *Dataset_1* after binary and static analysis results.

To avoid the curse of dimensionality problem, we grouped the OPCODEs in accordance with the MIPS instruction set [20].

We analyzed OPCODEs by dividing them into seven main groups. These are memory, branch, multiply, shifter, arithmetic, no operation, and other OPCODEs. When analyzing machine code, it is important to understand the specific behavior of OPCODE groups. For example, with the analysis of memory access-related OPCODEs, it can be determined how the program uses memory and manipulates data. Brief information about the OPCODE groups is given below.

- Memory access OPCODEs are used for read or write operations on the memory.
- Branch OPCODEs are used to transfer control to a different location in the code or to implement conditional statements and loops in a program.
- Multiply OPCODEs are used to perform multiplication operations in a program.
- Shifter OPCODEs are used to apply bitwise shift operations on data.
- Arithmetic OPCODEs are used to implement arithmetic operations in a program.

We also calculated the total and unique numbers of all OPCODEs, OPCODE groups, and NOP (No Operation) OPCODEs. Unique OPCODEs counts mean the number of individual OPCODEs. After all of the mentioned steps feature numbers of our dataset are reduced to 20. This dataset is labeled as *Dataset-2 (DS_2)*. The feature description of *Dataset_2* is given in Table I.

D. Classification

This research proposes a solution for a lightweight binary classification of ELF files created for IoT devices as malware and benignware. Prior to applying ML classifiers to our datasets, we conduct explanatory data analysis to reveal the connection between the features. After that to evaluate the performance of *Dataset_1* and *Dataset_2*, ML classifiers Logistic Regression, Naive Bayes, KNN, Decision Tree, SVM , and ANN are applied to the datasets. The results of this section are given in the next section.

IV. RESULTS

The main goal of this research is to reveal the impact of grouped OPCODE analysis. First, exploratory data analysis is conducted to achieve this goal. Then ML classifiers are applied to evaluate the performance of *Dataset_2*.

A. Exploratory Data Analysis

First, to discover patterns and anomalies in our *Dataset_2* the mean values of features are calculated. The mean values of the features of

TABLE I. FEATURE DESCRIPTION OF *DATASET_2*

Feature Name	Description
File Name	Name of Malware and Benignware files
Size	Size of the binary file of the software
Stripped	In case the software contains debugging information
Function Count	Number of functions available in the software
Label	Indicates whether the records in the dataset belong to malicious or normal software
NOP OPCODE Count	Number of NOP (No Operation) OPCODEs
OPCODE Count	Number of OPCODEs
Unique OPCODE Count	Number of unique OPCODEs
Memory Access OPCODE Count	Number of OPCODEs accessing memory
Unique Memory Access OPCODE Count	Number of unique OPCODEs accessing memory
Branch OPCODE Count	Number of OPCODEs that started executing another set of instructions under any condition.
Unique Branch OPCODE Count	Number of unique OPCODEs that start executing another set of instructions under any condition
Multiply OPCODE Count	Number of OPCODEs for multiplication and division operations
Unique Multiply OPCODE Count	Number of unique OPCODEs for multiplication and division operations
Shifter OPCODE Count	Number of OPCODEs for shift operations
Unique Shifter OPCODE Count	Number of unique OPCODEs for shift operations
Arithmetic OPCODE Count	Number of OPCODEs for arithmetic operations
Unique Arithmetic OPCODE Count	Number of unique OPCODEs for arithmetic operations
Arithmetic OPCODE Count	Number of OPCODEs excluding the specified groups
Unique Arithmetic OPCODE Count	Number of unique OPCODEs excluding the specified groups

TABLE II. MEAN VALUES OF MALWARE AND BENIGNWARE FEATURES

Feature Name	Mean Values of Malware	Mean Values of Benignware	Benignware to Malware Ratio
Size	87 475.59	147 766.2	1.69
Stripped	0.29	0.93	3.2
Function Count	65.55	228.97	3.50
NOP OPCODE Count	983.27	3811.18	3.87
OPCODE Count	7520.09	21 497.32	2.86
Unique OPCODE Count	38.27	40.92	1.07
Memory Access OPCODE Count	2621.75	7880.15	3.01
Unique Memory Access OPCODE Count	7.46	8.02	1.08
Branch OPCODE Count	1005.77	2729.19	2.71
Unique Branch OPCODE Count	8.75	11.02	1.26
Multiply OPCODE Count	43.61	70.52	1.62
Unique Multiply OPCODE Count	2.65	2.80	1.06
Shifter OPCODE Count	200.41	275.21	1.37
Unique Shifter OPCODE Count	4.09	4.10	1
Arithmetic OPCODE Count	1909.88	4524.67	2.37
Unique Arithmetic OPCODE Count	10.12	11.17	1.10
Arithmetic OPCODE Count	1738.64	6017.54	3.46
Unique Arithmetic OPCODE Count	5.18	3.78	0.73

benignware and malware are shown in Table II. As it is shown in Table II, most of the mean values of benignware features are higher than the mean values of malware. In the meantime, the mean values of "Unique Arithmetic Opcode" take attention with a higher value for malware. Besides, there is a significant difference in file size and the number of functions between benignware and malware. These values show us that the malware is quite small in size.

Second, Violin plots are used to present the distributions of unique OPCODE numbers in each group. Although the mean of the unique OPCODEs used in both malware and benignware seems close to each other, their distributions are quite different. It can be observed from the violin plots that for malware samples, the unique OPCODE numbers of each group are distributed around the median value, while it is more sparse for benignware samples. Also, the violin plot of unique multiply opcode takes attention with highly concentrated values around the median for malware and quite sparsely distributed values for benignware. The Violin Plots are shown in Fig. 3.

Lastly, the correlation matrix of the features in *Dataset_2* is created. It can be observed from the correlation matrix that none of the features has a high correlation with the label feature. This situation gives

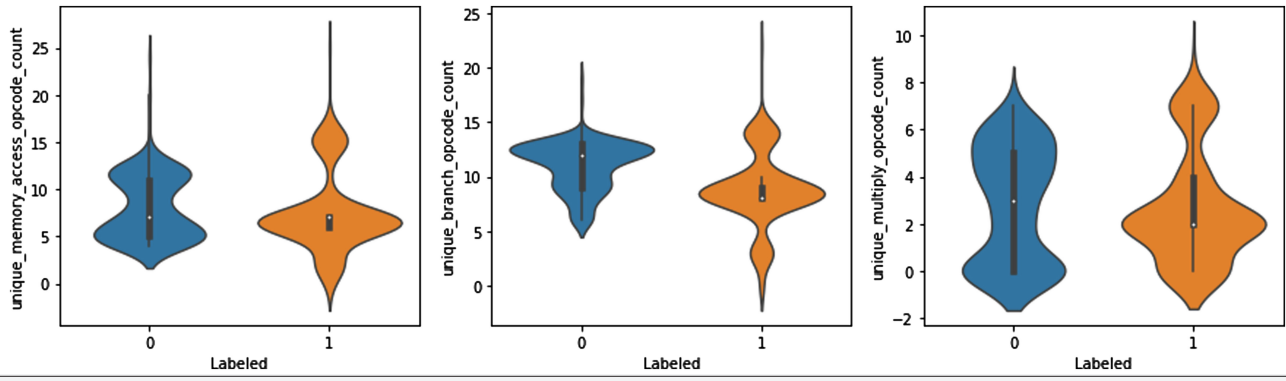

Fig. 3. Violin plots of unique OPCODE numbers in each group.

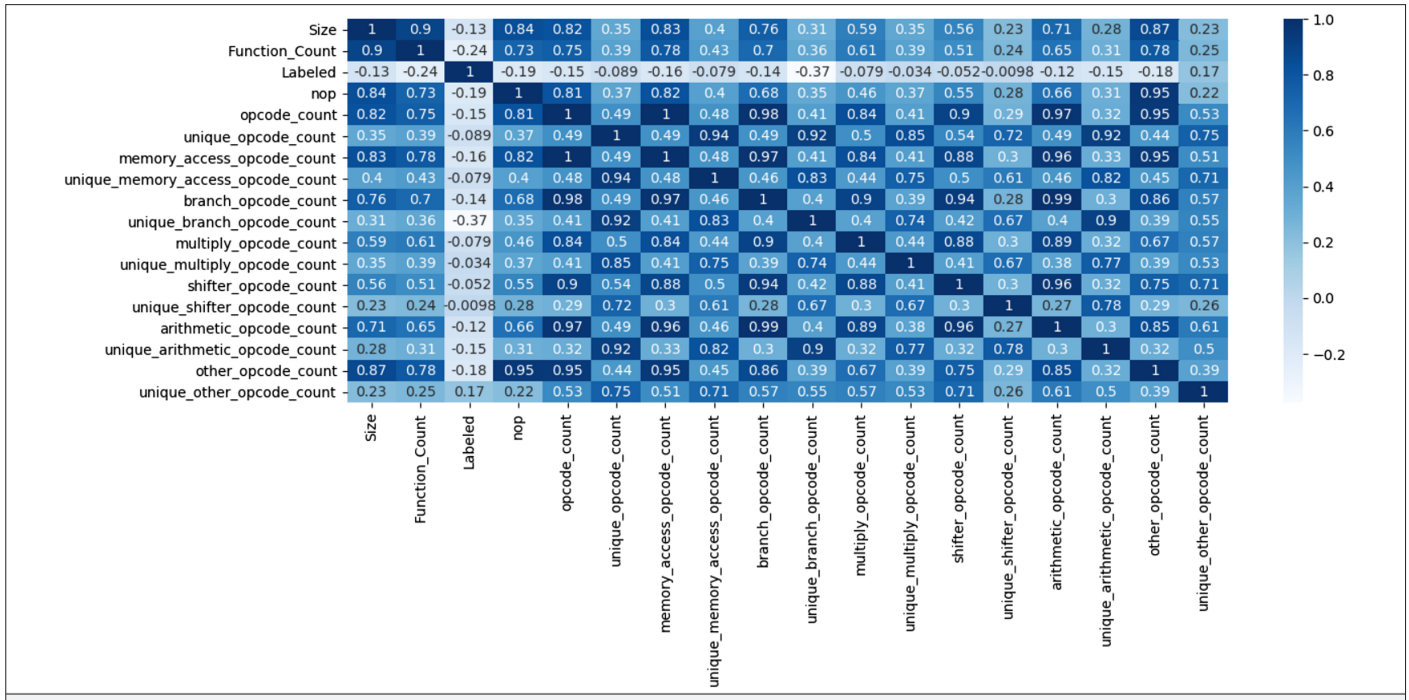

Fig. 4. Feature correlation matrix.

us insights into polynomial regressions, SVM, and decision tree classifiers that may perform better with *Dataset_2*. The feature correlation matrix is given in Fig. 4.

B. Evaluation of ML Classifiers

Six different supervised learning algorithms which are Logistic Regression, Naive Bayes, KNN, Decision Tree, SVM, and ANN classifiers are selected. These classifiers are trained with both *Dataset_1* and *Dataset_2* to reveal the impact of grouped OPCODE analysis. The datasets are separated into 80% for training and 20% for testing phases of classification. The general ML evaluation metrics of Accuracy, Precision, Recall, and F1 Score along with program runtime value are used to present the comparison of both classifiers and the datasets.

Except for the Naïve Bayes classifier, all other classifiers performed well with both of the datasets. Naïve Bayes depends on the probability of each feature belonging to a malware sample or not. Since none of the features in the datasets has a high correlation with the label data this is the expected performance from Naïve Bayes. For all other

TABLE III. CONFUSION MATRIX OF THE CLASSIFIERS WITH *DATASET_2*

A Logistic Regression

ACTUALS \ PREDICTIONS	0	1	
0	41	1	0
1	4	37	1
	0	1	

B Naïve Bayes

ACTUALS \ PREDICTIONS	0	1	
0	9	30	0
1	2	42	1
	0	1	

C KNN

ACTUALS \ PREDICTIONS	0	1	
0	39	1	0
1	5	38	1
	0	1	

D Decision Tree

ACTUALS \ PREDICTIONS	0	1	
0	41	0	0
1	0	42	1
	0	1	

E SVM

ACTUALS \ PREDICTIONS	0	1	
0	41	0	0
1	0	42	1
	0	1	

TABLE IV. EVALUATION OF CLASSIFIERS AND DATASETS

	Accuracy		Precision		Recall		F1 Score		Runtime	
Classification Algorithms	**DS_1**	**DS_2**	**DS_1**	**DS_2**	**DS_1**	**DS_2**	**DS_1**	**DS_2**	**DS_1**	**DS_2**
Logistic Regression	0.976	0.940	0.976	0.974	0.976	0.902	0.976	0.937	2.16	0.30
Naïve Bayes	0.646	0.614	0.607	0.583	0.829	0.955	0.701	0.724	1.29	0.21
KNN	0.976	0.928	0.976	0.974	0.976	0.884	0.976	0.927	1.34	0.24
Decision Tree	0.988	1	1	1	0.976	1	0.988	1	1.16	0.20
SVM	0.988	1	1	1	0.976	1	0.988	1	1.53	0.27
ANN	0.914	0.926	–	–	–	–	–	–	10.70	1.5

ANN, artificial neural network; SVM, support vector machine.

classifiers, both of the datasets have obtained high-performance values. The remarkable result is decreasing the runtime of classifiers when trained with *Dataset_2*. Logistic Regression is 7.2 times, KNN and SVM are 5.6 times, Decision Tree is 5.8 times, and ANN is 7.1 times faster when they are trained by *Dataset_2*. The confusion matrix of the classifiers are given in Table III and the evaluation of the classifiers is shown in Table IV.

V. CONCLUSIONS

The recent attacks revealed the severity of the malware threat for IoT devices. Unfortunately, the constrained nature of these devices makes it impossible to use high-resource-demanding anti-malware tools. There is an enormous need for efficient and lightweight anti-malware solutions. This study presents the preliminary results of our research for developing a lightweight anti-malware solution for IoT devices and also aims to reveal the impact of grouped OPCODE analysis. This study proposes a new malware classification methodology by grouping OPCODEs in accordance with the MIPS instruction set. By doing that the dataset feature number is reduced from 145 to 20. This feature reduction leverages the runtime. The ML classifiers from 5.6 to 7.2 times faster with our proposed methodology.

This study only covers malware and benignware compiled for MIPS architecture. Also, only static analysis features are used in the scope of this research. We believe that combining static and dynamic analysis features may leverage the performance of IoT malware detection. For this reason, developing a cross-architecture anti-malware tool for IoT devices that utilizes both static and dynamic analysis is determined as a future work of this research.

Peer-review: Externally peer-reviewed.

Author Contributions: Concept – I.G., H.H.K.; Design – I.G., H.H.K.; Supervision – A.H.Z., M.A.A.; Resources – I.G.; Materials – I.G.; Data Collection and/or Processing – I.G., H.H.K.; Analysis and/or Interpretation – I.G.; Literature Search – I.G.; Writing – I.G., H.H.K.; Critical Review – A.H.Z., M.A.A.

Declaration of Interests: The authors have no conflict of interest to declare.

Funding: This research is supported by EUREKA Cluster CELCTIC-NEXT under Project iCare4NextG, and in part by the Scientific and Technological Research Council of Turkey (TUBITAK).

REFERENCES

1. SonicWall Inc., *Mid-year Update: 2022 SonicWALL Cyber Threat Report* [Online]. Milpitas, CA USA: SonicWall Inc., Tech. Rep., 2022. Available: https://www.sonicwall.com/2022-cyber-threat-report/. [Accessed: 04 March 2023].
2. Zscaler, "IoT in the Enterprise: Empty office edition: What happens when employees abandon their smart devices at work?," [Online] 2021. Available: https://www.zscaler.com/resources/industry-reports/threatlabz-iot-in-the-enterprise.pdf. [Accessed: 04 March 2023].
3. M. Antonakakis et al., "Understanding the Mirai botnet," in 26th USENIX Security Symposium ({USENIX} Security 17), 2017, pp. 1093–1110, Vancouver, Canada.
4. I. Gulatas, H. H. Kilinc, A. H. Zaim, and M. A. Aydin, "Malware threat on edge/fog computing environments from Internet of things devices perspective," *IEEE Access*, vol. 11, pp. 33584–33606, 2023. [CrossRef]
5. Q. D. Ngo, H. T. Nguyen, V. Le, and D. H. Nguyen, "A survey of IoT malware and detection methods based on static features," *ICT Express*, vol. 6, no. 4, pp. 280–286, 2020. [CrossRef]
6. H. Darabian, A. Dehghantanha, S. Hashemi, S. Homayoun, and K. R. Choo, "An opcode-based technique for polymorphic Internet of things malware detection," *Concurrency Comput. Pract. Experience*, vol. 32, no. 6, p. e5173, 2020. [CrossRef]
7. J. Su, V. Danilo Vasconcellos, S. Prasad, S. Daniele, Y. Feng, and K. Sakurai, "Lightweight classification of IoT malware based on image recognition," in IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC), IEEE, vol. 2, 2018, pp. 664–669. [CrossRef]
8. A. Azmoodeh, A. Dehghantanha, and K.-K. R. Choo, "Robust malware detection for Internet of (battlefield) things devices using deep eigenspace learning," *IEEE Trans. Sustain. Comput.*, vol. 4, no. 1, pp.88–95, 2018. [CrossRef]
9. H. HaddadPajouh, A. Dehghantanha, R. Khayami, and K. R. Choo, "A deep recurrent neural network based approach for Internet of things malware threat hunting," *Future Gener. Comput. Syst.*, vol. 85, pp. 88–96, 2018. [CrossRef]
10. H. Haddadpajouh, A. Mohtadi, A. Dehghantanaha, H. Karimipour, X. Lin, and K. R. Choo, "A multikernel and metaheuristic feature selection approach for IoT malware threat hunting in the edge layer," *IEEE Internet Things J.*, vol. 8, no. 6, pp. 4540–4547, 2021. [CrossRef]
11. H. Alasmary *et al.*, "Analyzing and detecting emerging Internet of things malware: A graph-based approach," *IEEE Internet Things J.*, vol. 6, no. 5, pp. 8977–8988, 2019. [CrossRef]
12. Y. Meidan *et al.*, "N-baiot—Network-based detection of IoT botnet attacks using deep autoencoders," *IEEE Pervasive Comput.*, vol. 17, no. 3, pp. 12–22, 2018. [CrossRef]
13. J. Jeon, J. H. Park, and Y.-S. Jeong, "Dynamic analysis for IoT malware detection with convolution neural network model," *IEEE Access*, vol. 8, pp. 96899–96911, 2020. [CrossRef]
14. V. Rey, P. M. Sánchez Sánchez, A. Huertas Celdrán, and G. Bovet, "Federated learning for malware detection in IoT devices," *Comput. Netw.*, vol. 204, p. 108693, 2022. [CrossRef]

15. Y. Pa, S. Suzuki, K. Yoshioka, T. Matsumoto, T. Kasama, and C. Rossow, "IoTPOT: Analysing the rise of IoT compromises," in 9th USENIX Workshop on Offensive Technologies (WOOT 15), 2015, Denver, Colorado.
16. S. Garcia, A. Parmisano, J. Erquiaga, and Maria, "Iot-23: A labeled dataset with malicious and benign IoT network traffic (version 1.0.0)," [Online] 2020. Available: https://www.stratosphereips.org/datasets-iot23.
17. M. Safaei Pour *et al.*, "On data-driven curation, learning, and analysis for inferring evolving Internet-Of-Things (IoT) botnets in the wild," *Comput. Sec.*, vol. 91, p. 101707, 2020. [CrossRef]
18. T. Trajanovski, and N. Zhang, "An automated and comprehensive framework for IoT botnet detection and analysis (IoT-BDA)," *IEEE Access*, vol. 9, pp. 124360–124383, 2021. [CrossRef]
19. "Malware bazaar," *Malware Database*. Available: https://bazaar.abuse.ch/. [Accessed: 05 March 2023].
20. "MIPS instruction set" [Online]. Available: https://www.mips.com/products/architectures/mips32-2/. [Accessed: 05 March 2023].

Ibrahım Gulatas received a B.S. degree in Computer Engineering from Turkish Naval Academy, Turkey, in 2010, and a M.S. degree in Computer Engineering from Bahcesehir University, Turkey, in 2018. Currently, he is a Ph.D. Candidate in Istanbul Commerce University, Turkey. His current research interests include information security and malware analysis.

Hacı Hakan Kılınc is working as an Innovation Project Manager for Orion Innovation Turkey since December 2019. He worked as a cybersecurity product line manager for Netas between 2014 and 2019. He received his B.S. degree in Mathematics and Computer Science from Aegean University, Izmir, Turkey, in 1997. He received his M.S. degree in 2001 in the Department of Computer Engineering at the Izmir Institute of Technology. He worked as a visiting scholar at the University of Texas at Dallas between 2009 and 2011. He holds a Ph.D. about the security of SIP (Session Initiation Protocol) from the Department of Electronics Engineering at the Gebze Technical University in 2014.

Muhammed Alı Aydın is currently working as an Associate Professor at the Computer Engineering Department of Istanbul University–Cerrahpasia. His interest mainly focuses on Cyber Security, Cryptography, Network Security, and Communication–Network Protocols. He received his Ph.D. degree in Computer Engineering from Istanbul University and he has completed post-doctoral research at Telecom SudParis in the Department of Computer Science.

Abdul Halım Zaım is currently a faculty member in the Department of Computer Engineering at the Faculty of Engineering at Istanbul Commerce University, and the Director of the Center for Information Technology Application and Research at Istanbul Commerce University. Abdul Halim Zaim has been serving as Director of the Technology Transfer Office. His research interest mainly focuses on IoT, Big Data, Network Design, Cyber Security, Network Security, and Communication–Network Protocols. He received his MS degree in Computer Engineering from Bogazici University in 1996 and his Ph.D. in Electrical and Computer Engineering from North Carolina State University (NCSU) in 2001.